FCKeditor 使用指南

作者：何明旺

目　　录

1 FCKeditor 的下载及介绍

1.1 下载地址

FCKeditor主页： http://www.FCKeditor.net

演示地址：http://www.FCKeditor.net/demo

文档位置：http://docs.FCKeditor.net

下载地址：http://www.FCKeditor.net/download

1.2 FCKeditor 下载包的介绍

FCKeditor_2.6.3.zip 是 Js 写的一个客户端

FCKeditor-java-2.4-bin.zip 用于服务端的，可以实现文件上传等

FCKeditor-java-demo-2.4.war 一个演示例子

2 FCKeditor 的目录和文件精简

因为这个编辑器是支持多语言的，所以首先我们针对使用对其做相应的冗余文件删除。

1. 临时文件及文件夹删除：从根目录下开始删除一切以“_”开头的文件及文件夹，因为他们为临时文件和文件夹。删除这类临时文件及文件夹之后，我们还 要删除一些根目录下的多余文件，根目录下我们只保留 fckconfig.js（配置文件）、fckeditor.js（js 方式调用文件）、 fckstyles.xml（样式）、fck templates.xml（模板）文件和 editor 文件夹。
2. 将 editor/filemanager/upload 目录下文件及文件夹清空
3. editor\filemanager\browser\default\connectors 目录：存放编辑器所支持的 Web 动态语言，test.html 文件可以帮助你查看某语言下的上传设置等（具体上传设置我将在后面的配置作较为详细讲解），可以自行决定是否删除。
4. editor\lang 目录：存放的是多语言配置文件，因为我们只可能用到 en 和 zh-cn（简体中文）所以，根据我的选择，我删掉其他的语言配置文件。
5. editor\skins 界面目录：默认带有三个界面（default：默认界面，加载速度相对较快；office2003：相对 pp 的界面，不过速度确实要慢些；silver：银白色界面，加载速度也相对较快），可以自行决定是否删除其中一两个。

到此精简完成，你会发现整个编辑器确实“瘦身”不少。

3 在页面创建 FCKeditor

3.1 Js 创建 FCKeditor 实例：

在对应对应的 Html 引入 FCKeditor.js

```
<script type="text/javascript" src="fckeditor/fckeditor.js"></script>
```

3.1.1 方法一：内嵌方法(推荐)

在需要嵌入 FCKeditor 的地方写上如下这段 Js：

```
<script type="text/javascript">
  var oFCKeditor = new FCKeditor('FCKeditor1');
```

```
oFCKeditor.BasePath = "/project/fckeditor/";
oFCKeditor.Create();
</script>
```

例如：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>FCKeditor - Sample</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8">
        <meta name="robots" content="noindex, nofollow">
        <script type="text/javascript"
src="fckeditor/fckeditor.js"></script>
    </head>
    <body>
        <form>
            <script type="text/javascript">
                var oFCKeditor = new FCKeditor('FCKeditor1');
                oFCKeditor.BasePath = "/project/fckeditor/"; // BasePath
属性必须设置正确，否则会出现404错误，必须以“/”结尾
                oFCKeditor.Create();
            </script>
        </form>
    </body>
</html>
```

3.1.2 方法二：替换页面中的 Textarea

先在页面定一个 Textarea(必须为 name 属性指定值) 再在 Html 写入这么一段 Js 代码：

```
<script type="text/javascript">
  window.onload = function()
  {
  var oFCKeditor = new FCKeditor( 'MyTextarea' ) ;
  oFCKeditor.BasePath = "/project/fckeditor/" ;
  oFCKeditor.ReplaceTextarea() ;
  }
</script>
```

Html 的 Body 中：

```
    <textarea id="MyTextarea" name="MyTextarea">This is <b>the</b> initial value.</textarea>
```

例如：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
    <head>
        <title>FCKeditor - Sample</title>
        <meta http-equiv="Content-Type" content="text/html;
```

```
charset=utf-8">
        <meta name="robots" content="noindex, nofollow">
        <script type="text/javascript"
src="fckeditor/fckeditor.js"></script>
        <script type="text/javascript">
        window.onload = function() {
            var oFCKeditor = new FCKeditor('MyTextarea');
            oFCKeditor.BasePath = "/project/fckeditor/"; // BasePath属性
必须设置正确，否则会出现404错误，必须以“/”结尾
            oFCKeditor.ReplaceTextarea();
        }
        </script>
    </head>
    <body>
        <textarea id="MyTextarea" name="MyTextarea">This is <b>the</b>
initial value.</textarea>
    </body>
</html>
```

3.1.3 方法三：适合于 Ajax 的调用方法

```
<script type="text/javascript">
    //使用getElementById方法取得myFCKeditor ID元素
    var div = document.getElementById("myFCKeditor");
    //创建fckeditor实例
    var fck = new FCKeditor("myFCKeditor");
    //使用innerHTML方法，在myFCKeditor div元素里创建编辑器
    div.innerHTML = fck.CreateHtml();
</script>
```

3.1.4 Js 中 FCKeditor 对象的属性(集合)和方法

3.1.4.1属性

属性名	描述	默认值
Width	宽度	100%
Height	高度	200
Value	初始化内容	（空字符串）
ToolbarSet	工具条的集合名的称(内置有 Default 和 Basic，也可自己定制)	Default
BasePath	编辑器的基本路径	/fckeditor/
CheckBrowser	是否在显示编辑器前检查浏览器兼容性	true
DisplayErrors	是否显示提示错误	true
只读属性名	描述	数据类型
Description	描述	string
EditMode	编辑状态	Integer

Name	名字	string
Status	状态	Integer

3.1.4.2集合

Config[Key]=value;
这个集合用于更改配置中某一项的值,如：oFckeditor.Config["DefaultLanguage"]="zh-cn";

3.1.4.3方法

```
function AttachToOnSelectionChange(functionPointer)

function CleanAndPaste(html)

function CreateElement(tag)

function CreateLink(url)

function ExecOnSelectionChange() //Fires OnSelectionChange event in
event manager

function ExecOnSelectionChangeTimer()

function ExecuteNamedCommand(commandName, commandParameter)

function ExecuteRedirectedNamedCommand(commandName, commandParameter)

function Focus()

function GetHTML(format) // doesnt work. Use GetXHTML instead.

function GetNamedCommandState(commandName)

function GetNamedCommandValue(commandName)

function GetXHTML(format)

function InitializeBehaviors()

function InsertElement(element)

function InsertElementAndGetIt(e)

function InsertHtml(html)

function IsDirty();
```

```
function MakeEditable()

function OnDoubleClick(element)

function Paste()

function PasteAsPlainText()

function PasteFromWord()

function Preview()

function RegisterDoubleClickHandler(handlerFunction, tag)

function ResetIsDirty();

function SetHTML(html, forceWYSIWYG)

function SetStatus()

function ShowContextMenu(x, y)

function SwitchEditMode()

function UpdateLinkedField()
```

EventsOnce the editor loading is complete and it is ready to use (and interact with JavaScript), a standard function is called in the page that contains the editor, if the function is defined.

This function must be named "FCKeditor_OnComplete" and receives the related editor instance as the parameter. Using it, you can execute any initial code that makes the initial interaction with the editor.

This is a declaration example:

```
function FCKeditor_OnComplete( editorInstance ) {
    alert( editorInstance.Name ) ;
}
```

Apart the above standard event, every FCKeditor instance has a "Event" object that can be used to listen for events to be fired.

For example, the following code listens for the "OnSelectionChange" to execute custom code:

```
<script type="text/javascript">
var counter = 0 ;
function DoSomething( editorInstance ) {
    window.document.title = editorInstance.Name + ' : ' + ( ++counter ) ;
```

```
}

function FCKeditor_OnComplete( editorInstance ) {
    editorInstance.Events.AttachEvent( 'OnSelectionChange',
DoSomething ) ;
}
</script>
```

Note that every callback function receives the editor instance as a parameter.

The following is the list of events available:

OnSelectionChange: fired when the actual selection in the editor area changes (by selection I mean the cursor position too... it changes on key strokes). Note: In IE6, this event does not fire on every keystroke, but only on some random keystrokes. Handy!

OnAfterSetHTML: fired once the HTML is loaded in the editor (including when changing views).

OnStatusChange: fired when the editor status changes. The following constants are also available globally in the page: FCK_STATUS_NOTLOADED, FCK_STATUS_ACTIVE and FCK_STATUS_COMPLETE.

OnPaste: fired when something is pasted in the editor

3.1.5 FCKeditor 的 JS 构造器

Var FCKeditor = function(instanceName, width, height, toolbarSet, value)

- 其中 instanceName 为编辑器输出的 Textarea 元素的 name 属性值，必须指定
- 参数会赋给同名属性

3.1.6 将从后台读取的数据显示在 FCKeditor 中

如果想要使用从数据库读来的文本数据或者是后台来自文件的 txt/html 文本数据。只要在

<textarea name="content" cols="80" rows="4"></textarea>

中加入自己的显示内容的 formbean 对应字段即可

```
<textarea name="content" cols="80" rows="4">
    <c:out value="${contentData}" />
</textarea>
```

这样内容就会被显示在 FCKeditor 编辑框中了，点击提交按钮以后就可以在后台的相应 java action 中得到 content 参数中的内容就是页面上 FCKeditor 中的内容数据了。可以在 struts/jsf 中使用。

3.2 在 Jsp 中通过自定义标签创建：

可以参考：

- 演示工程：fckeditor-java-demo-2.4.war
- fckeditor-java-2.4-bin.zip 下面的文档

第一步：将相关的 Jar 包 Copy 到工程的 lib 目录下面(5 个)

第二步：在 Jsp 页面中引入自定义标签：

```
<%@ taglib uri="http://java.fckeditor.net" prefix="FCK" %>
```

第三步：在 Jsp 页面的 Body 中使用自定义标签：

```
<FCK:editor instanceName="EditorDefault" basePath=”/fckeditor” value=”this is content” />
```

也可以这样：

```
<FCK:editor instanceName="EditorDefault">
    <jsp:attribute name="value">This is some sample text</jsp:attribute>
</FCK:editor>
```

注意事项：

- basePath 属性值以“/”开头，这个“/”代表当前工程的根路径
- 一定要设置 value 属性值，并且值不能为空字符串（可以为” ”,中间有一个空格），否则会报空指针异常。

3.3 FCKeditor API 调用

第一步：在 Jsp 页面引入需要的 Java 类：<%@ page language="java" import="com.fredck.FCKeditor.*" %>

第二步：

```
<form action="show.jsp" method="post" target="_blank">
    <%
        FCKeditor oFCKeditor;
        oFCKeditor = new FCKeditor(request, "content");
        oFCKeditor.setBasePath("/TestFCKeditor/fckeditor/");
        oFCKeditor.setValue("init value");
        out.println(oFCKeditor);
    %>
    <br>
    <input type="submit" value="Submit">
</form>
```

3.4 适时打开编辑器

很多时候，我们在打开页面的时候不需要直接打开编辑器，而在用到的时候才打开，这样一来有很好的用户体验，另一方面可以消除 FCK 在加载时对页面打开速度的影响，点击“Open Editor"按钮后才打开编辑器界面。

实现原理：

使用 JAVASCRIPT 版的 FCK，在页面加载时（未打开 FCK），创建一个隐藏的 TextArea 域，这个 TextArea 的 name 和 ID 要和创建的 FCK 实例名称一致，然后点击"Open Editor"按钮时，通过调用一段函数，使用 FCK 的 ReplaceTextarea()方法来创建 FCKeditor，代码如下：

```
<script type="text/javascript">
    function showFCK() {
        var oFCKeditor = new FCKeditor('fbContent');
        oFCKeditor.BasePath = '/FCKeditor/';
        oFCKeditor.ToolbarSet = 'Basic';
        oFCKeditor.Width = '100%';
        oFCKeditor.Height = '200';
        oFCKeditor.ReplaceTextarea();
    }
</script>
<textarea name="fbContent" id="fbContent"></textarea>
```

4 修改 FCKeditor 的配置：

4.1 方法一：修改 fckconfig.js 文件

FCKeditor 的配置主要是通过 *fckeditor/fckconfig.js* 文件配置的，只需修改其中对应的属性值即可

4.2 方法二：使用一个额外的配置文件覆盖默认配置

第一步：在 WebRoot/ fckeditor/ 文件夹下(其他文件夹也可以)创建一个 *myfckconfig.js* 文件，并将要修改属性设置到此 Js 文件，如：

```
FCKConfig.AutoDetectLanguage    = false ;
FCKConfig.DefaultLanguage       = 'zh-cn' ;
```

第二步：指定自定义配置文件：

方法一： 在 fckconfig.js 中指定自定义配置文件所在的位置(应用到了所有的 FCKeditor)： **FCKConfig.CustomConfigurationsPath = FCKConfig.EditorPath + 'myfckconfig.js';**

方法二：在创建 FCKeditor 的时候为其指定对应的配置文件路径(只改变了当前 FCKeditor 实例的配置)：

```
<script type="text/javascript">
  var oFCKeditor = new FCKeditor('FCKeditor1');
  oFCKeditor.BasePath = "../../fckeditor/";
  oFCKeditor.Config["CustomConfigurationsPath"]= oFCKeditor.BasePath +" myfckconfig.js";
  oFCKeditor.Create();
</script>
```

4.3 配置的加载顺序

1. 加载主配置文件
2. 加在载自定义配置文件(如果有)，覆盖相同的配置项
3. 使用对应实例的配置覆盖相同的配置项(只对当前实例有效)

4.4 提示

1. 系统会自动侦测并并用适当的界面语言(默认，可以修改)
2. 不能删除主配置文件：fckconfig.js
3. 修改配置后要清空浏览器缓存，以免影响结果(或访问时强制刷新也可以：IE 中强制刷新的快捷键为：Ctrl + F5， FireFox 中强制刷新的快捷键为：Shift + Ctrl + R)

4.5 一般需要修改的配置项

4.5.1 默认语言

打开 fckconfig.js 文件（此文件是 utf-8 编码）， 找到 FCKConfig.AutoDetectLanguage = true; 此句作用为自动检测语言，默认为 true，即表示编辑器会根据系统语言自动检测加载相应的语言，我们可以将其改为 false，不让其检测，然后 将 FCKConfig.DefaultLanguage = 'en';改为简体中文"zh-cn"。

4.5.2 自定义 ToolbarSet， 去掉一些不需要的功能

说明：

```
FCKConfig.ToolbarSets["Default"] = [
    ['Source','DocProps','-','Save','NewPage','Preview','-','Templates'],
    ['Cut','Copy','Paste','PasteText','PasteWord','-','Print','SpellCheck'],
    ['Undo','Redo','-','Find','Replace','-','SelectAll','RemoveFormat'],
    ['Form','Checkbox','Radio','TextField','Textarea','Select','Button','ImageButton','HiddenField'],
    '/',
    ['Bold','Italic','Underline','StrikeThrough','-','Subscript','Superscript'],
    ['OrderedList','UnorderedList','-','Outdent','Indent','Blockquote'],
    ['JustifyLeft','JustifyCenter','JustifyRight','JustifyFull'],
    ['Link','Unlink','Anchor'],
    ['Image','Flash','Table','Rule','Smiley','SpecialChar','PageBreak'],
    '/',
    ['Style','FontFormat','FontName','FontSize'],
    ['TextColor','BGColor'],
    ['FitWindow','ShowBlocks','-','About']    // No comma for the last row.
] ;
```

其中[]中为一组工具按钮集，之间用“,”连接，这个“,”将显示为一个虚线分隔符，[]中的“'-'”将显示为分隔符，[]与[]之间的“'/'”代表换行，将使前后两个工具按钮组分别显示在上下两行。

第一步：在自定义的 Js 配置文件中写上你要的工具按钮集，如：

```
FCKConfig.ToolbarSets["CustomToolbar"] = [
    ['Source','DocProps','-','Save','NewPage','Preview','-','Templates'],
    ['Undo','Redo','-','Find','Replace','-','SelectAll','RemoveFormat'],
    '/',
    ['Style','FontFormat','FontName','FontSize'],
    ['TextColor','BGColor'],
    ['FitWindow','ShowBlocks']
] ;
```

对于上面的配置选项的汇总如下表：

代码名称	功能	代码名称	功能
Source	源代码	DocProps	页面属性

-	丨分隔符	Save	保存
NewPage	新建	Preview	预览
Templates	模板	Cut	剪切
Copy	复制	Paste	粘贴
PasteText	粘贴为无格式文本	PasteWord	从 MS Word 粘贴
Print	打印	SpellCheck	拼写检查
Undo	撤消	Redo	重做
Find	查找	Replace	替换
SelectAll	全选	RemoveFormat	清除格式
Form	表单	Checkbox	复选框
Radio	单选框	TextField	单行文本
Textarea	多行文本	Select	列表菜单
Button	按钮	ImageButton	图像域
HiddenField	隐藏域	Bold	加粗
Italic	倾斜	Underline	下划线
StrikeThrough	删除线	Subscript	下标
Superscript	上标	OrderedList	插入／删除编号列表
UnorderedList	插入／删除项目列表	Outdent	减少缩进
Indent	增加缩进	JustifyLeft	左对齐
JustifyCenter	居中对齐	JustifyRight	右对齐
JustifyFull	两端对齐	Link	插入／编辑链接
Unlink	取消链接	Anchor	插入／编辑锚点链接
Image	插入编辑图像	Flash	插入／编辑 Flash
Table	插入／编辑表格	Rule	插入水平线
Smiley	插入表情	SpecialChar	插入特殊符号
PageBreak	插入分页	Style	样式
FontFormat	格式	FontName	字体
FontSize	大小	TextColor	文本颜色
BGColor	背景颜色	FitWindow	全屏编辑
About	关于 Fuckeditor		

4.5.3 加上几种常用的字体

将配置文件中的 FCKConfig.FontNames 属性的值修改(各字体名称之间用“;”分隔开)，如：

FCKConfig.FontNames = '宋体;隶书;楷体_GB2312;黑体;华文行楷;Arial;Comic Sans MS;Courier New;Tahoma;Times New Roman;Verdana' ;

4.5.4 修改“回车” 和 “Shift + 回车”的换行行为

修改 FCKConfig.EnterMode 和 FCKConfig.ShiftEnterMode 的值(可为 p、div 或 br)，如下为按 回车 的时候换行，按 Shift + 回车 的时候为增加一个段落：

FCKConfig.EnterMode = 'br' ;

FCKConfig.ShiftEnterMode = 'p' ;

4.5.5 修改编辑区的样式文件

将你需要的样式添加到 WebRoot\fckeditor\editor\css\fck_editorarea.css'；文件的后面即可。

4.5.6 更换表情图片

```
//表情图片所在文件夹的路径
FCKConfig.SmileyPath = FCKConfig.BasePath + 'images/smiley/msn/';
//需要显示的表情图片名称（之间用“,”隔开）
FCKConfig.SmileyImages  =
['regular_smile.gif','sad_smile.gif','wink_smile.gif','teeth_smile.gi
f','confused_smile.gif','tounge_smile.gif','embaressed_smile.gif','om
g_smile.gif','whatchutalkingabout_smile.gif','angry_smile.gif','angel
_smile.gif','shades_smile.gif','devil_smile.gif','cry_smile.gif','lig
htbulb.gif','thumbs_down.gif','thumbs_up.gif','heart.gif','broken_hea
rt.gif','kiss.gif','envelope.gif'] ;
//每行显示的表情图片个数
FCKConfig.SmileyColumns = 8 ;
//表情图片弹出页面的宽度
FCKConfig.SmileyWindowWidth    = 320 ;
//表情图片弹出页面的高度
FCKConfig.SmileyWindowHeight   = 210 ;
```

注：

如果因为表情图片太大或者因为表情图片过多导出弹出页面尺寸过大，可通过修改 fckeditor\editor\dialog\fck_smiley.html 页面解决：

1. 将 Head 部分

```
window.onload = function ()
{
// First of all, translate the dialog box texts
oEditor.FCKLanguageManager.TranslatePage(document) ;
dialog.SetAutoSize( true ) ;
}
```

中的 dialog.SetAutoSize(true) ; 注释掉，此句作用为自动将弹出页面尺寸调整至适合大小。

2. 将 <body style="overflow: hidden"> 中的 style="overflow: hidden" 修改为 style="overflow: auto" 至于 overflow 样式的作用可参考 CSS 的相关文档。

4.5.7 编辑区域的右键菜单功能

```
FCKConfig.ContextMenu =
['Generic','Link','Anchor','Image','Flash','Select','Textarea','Check
box','Radio','TextField','HiddenField','ImageButton','Button','Bullet
edList','NumberedList','Table','Form','DivContainer'] ;
```

4.6 fckconfig.js 配置参数选项说明

FCKConfig.EditorPath	fckeditotr 文件夹所在的路径
FCKConfig.CustomConfigurationsPath	自定义配置文件路径和名称
FCKConfig. EditorAreaCss	编辑区的样式表文件
FCKConfig.EditorAreaStyles	编辑区的样式表风格
FCKConfig.ToolbarComboPreviewCSS	工具栏预览 CSS
FCKConfig.DocType	文档类型
FCKConfig.BaseHref	相对链接的基地址
FCKConfig.FullPage=true/false	是否允许编辑整个 HTML 文件,还是仅允许编辑 BODY 间的内容
FCKConfig.StartupShowBlocks	决定是否启用"显示模块"
FCKConfig.Debug	是否开启调试功能,这样,当调用 FCKDebug.Output() 时,会在调试窗中输出内容
FCKConfig.SkinPath	设置默认皮肤路径
FCKConfig.PreloadImages	预装入的图片
FCKConfig.PluginsPath	插件文件夹
FCKConfig.AutoDetectLanguage	自动语言检查
FCKConfig.DefaultLanguage	默认语言设计，建议改成 zh-cn
FCKConfig.ContentLangDirection	默认文字方向(ltr/rtl)
FCKConfig.ProcessHTMLEntities	处理 HTML 实体
FCKConfig.IncludeLatinEntities	包括拉丁文
FCKConfig.IncludeGreekEntities	包括希腊文
FCKConfig.ProcessNumericEntities	处理数字实体
FCKConfig.AdditionalNumericEntities	附加的数字实体
FCKConfig.FillEmptyBlocks	使用这个功能,可以将空的块级元素用空格来替代,是否填充空块
FCKConfig.FormatSource	在切换到代码视图时是否自动格式化代码
FCKConfig.FormatOutput	当输出内容时是否自动格式化代码
FCKConfig.FormatIndentator	当在源码格式下缩进代码使用的字符
FCKConfig.StartupFocus	开启时 FOCUS 到编辑器
FCKConfig.ForcePasteAsPlainText	强制粘贴为纯文本
FCKConfig.AutoDetectPasteFromWord	是否自动探测从 word 粘贴文件,仅支持 IE
FCKConfig.ShowDropDialog	是否显示下拉菜单
FCKConfig.ForceSimpleAmpersand	是否不把&符号转换为 XML 实体
FCKConfig.TabSpaces	TAB 键产生的空格字符数
FCKConfig.ShowBorders	合并边框
FCKConfig.SourcePopup	弹出
FCKConfig.ToolbarStartExpanded	当页面载入的时候，工具栏默认情况下是否展开
FCKConfig.ToolbarCanCollapse	是否允许展开折叠工具栏
FCKConfig.IgnoreEmptyParagraphValue	是否忽略空的段落值
FCKConfig.FloatingPanelsZIndex	浮动面板索引
FCKConfig.HtmlEncodeOutput	是否将 HTML 编码输出

FCKConfig.TemplateReplaceAll	是否替换所有模板
FCKConfig.TemplateReplaceCheckbox	是否可用模板替换多选框
FCKConfig.ToolbarLocation	工具栏位置
FCKConfig.ToolbarSets	允许使用 TOOLBAR 集合
FCKConfig.EnterMode	编辑器中直接回车，在代码中生成，可选为 p | div | br
FCKConfig.ShiftEnterMode	编辑器中 Shift+回车，在代码中生成，可选为 p | div | br
FCKConfig.ContextMenu	右键菜单内容
FCKConfig.FontColors	文字颜色列表
FCKConfig.FontFormats	文字格式列表
FCKConfig.FontNames	字体列表，可加入国内常用的字体，如宋体、楷体、黑体等
FCKConfig.FontSizes	字号列表
FCKConfig.StylesXmlPath	指定样式 XML 文件路径
FCKConfig.TemplatesXmlPath	指定模板 XML 文件路径
FCKConfig.SpellChecker	设置拼写检查器
FCKConfig.IeSpellDownloadUrl	下载拼写检查器的网址
FCKConfig.FirefoxSpellChecker	火狐浏览器下的拼写检查器
FCKConfig.MaxUndoLevels	最多可以撤销多少步
FCKConfig.DisableObjectResizing	选定编辑区的大小是否可变
FCKConfig.TemplatesXmlPath	模板的 XML 文件路径
FCKConfig.BodyId	设置编辑器的 id
FCKConfig.BodyClass	设置编辑器的 class
FCKConfig.DefaultLinkTarget	设置链接默认情况下的 target 属性
FCKConfig.LinkBrowser	是否允许在插入链接时浏览服务器
FCKConfig.LinkBrowserURL	插入链接时浏览服务器的 URL
FCKConfig.LinkBrowserWindowWidth	链接目标浏览器窗口宽度
FCKConfig.LinkBrowserWindowHeight	链接目标浏览器窗口高度
FCKConfig.ImageBrowser	插入图片时是否允许浏览服务器功能
FCKConfig.ImageBrowserURL	浏览服务器时运行的 URL
FCKConfig.ImageBrowserWindowWidth	图像浏览器窗口宽度
FCKConfig.ImageBrowserWindowHeight	图像浏览器窗口高度
FCKConfig.FlashBrowser	参照 FCKConfig.Imag…
FCKConfig.FlashBrowserURL	
FCKConfig.FlashBrowserWindowWidth	
FCKConfig.FlashBrowserWindowHeight	
FCKConfig.LinkUpload	是否开启文件上传功能
FCKConfig.LinkUploadURL	上传的文件的保存路径
FCKConfig.LinkUploadAllowedExtensions	允许上传的文件类型列表
FCKConfig.LinkUploadDeniedExtensions	拒绝上传的文件类型列表
FCKConfig.ImageUpload	参照 FCKConfig.LinkUpload…
FCKConfig.ImageUploadURL	
FCKConfig.ImageUploadAllowedExtensions	

FCKConfig.ImageUploadDeniedExtensions	
FCKConfig.FlashUpload	参照 FCKConfig.LinkUpload…
FCKConfig.FlashUploadURL	
FCKConfig.FlashUploadAllowedExtensions	
FCKConfig.FlashUploadDeniedExtensions	
FCKConfig.SmileyPath	表情文件路径，您可以设置此项更改表情
FCKConfig.SmileyImages	表情文件
FCKConfig.SmileyColumns	将表情分成几列显示
FCKConfig.SmileyWindowWidth	显示表情窗口的宽度，单位像素
FCKConfig.SmileyWindowHeight	显示表情窗口的高度，单位像素

4.7 自定义工具栏按钮

需要修改的文件：

1. fckeditor/fckconfig.js
2. fckeditor/editor/lang/zh-cn.js
3. fckeditor/editor/_source/internals/fckregexlib.js
4. fckeditor/editor/_source/internals/fckcommands.js
5. fckeditor/editor/_source/internals/fcktoolbaritems.js

具体步骤：

第一步：修改 fckconfig.js 文件

1. 我们找到 FCKConfig.ToolbarSets["Default"]这一行，在最后即'About'后添加一个工具按钮 MyToolBar，名称为 MyToolBar；
2. 找到 FCKConfig.DefaultLanguage，修改语言为：zh-cn；
3. 找到 FCKConfig.AutoDetectLanguage，设置为 false，即关闭语言的自动检测功能；

第二步：修改 zh-cn.js 文件

1. 在最后加入：**MyToolBar : "我的自定义工具栏"**
2. 注意它前面的一个最后要加多一个逗号；

第三步：修改 fckregexlib.js 文件

找到 NamedCommands 这一行，在最后加入：MyToolBar

第四步：修改 fckcommands.js 文件

找到 FCKCommands.GetCommand 函数，在其中加入：

```
    case 'MyToolBar':
        oCommand = new FCKDialogCommand('MyToolBar', FCKLang.MyToolBar,
'dialog/MyToolBar.html', 450, 400 );
        break ;
```

第五步：修改 fcktoolbaritems.js 文件

找到 FCKToolbarItems.GetItem 函数，在其中加入：

```
    case 'MyToolBar':
       oItem = new FCKToolbarButton('MyToolBar', FCKLang.MyToolBar, null,
null, null, true, 72);
        break ;
```

这里 72 是表示 skins 目录下各个皮肤目录中 fck_strip.gif 图片文件中的图片索引，我们这里用和命令 ShowBlocks 一样的图标（一个问号图片）。

第六步：在目录 fckeditor/editor/dialog/ 下面创建一个 MyToolBar.html 文件，内容如下：

```
<html>
    <head>
        <title>MyTool Dialog Page</title>
        <meta http-equiv="Content-Type" content="text/html;
charset=utf-8" />
        <meta content="noindex, nofollow" name="robots" />
        <script src="common/fck_dialog_common.js"
type="text/javascript"></script>
        <script type="text/javascript">
            var oEditor = window.parent.InnerDialogLoaded();
            window.onload = function() {
                window.parent.SetOkButton(true);
                window.parent.SetAutoSize(true);
            }

            function Ok() {
                var oActiveEl =
oEditor.FCK.EditorDocument.createElement('SPAN');
                oActiveEl.innerHTML = GetE('txtName').value;
                oEditor.FCKUndo.SaveUndoStep();
                oActiveEl = oEditor.FCK.InsertElement(oActiveEl);
                return true;
            }
        </script>
    </head>
    <body style="overflow: hidden">
        <table width="100%" style="height: 100%">
            <tr>
                <td align="center">
                    请输入文字：
                    <input id="txtName" type="text" />
                </td>
            </tr>
        </table>
    </body>
</html>
```

4.8 自定义右键菜单

1. 首先，在 editor/lang/zh-cn.js 里添加你所要添加的工具条 item 的名字，此文件定义了一个 FCKLang 变量，仿照里边的格式写哦，如:**testComman: "测试菜单",**注意，如果是不在倒数第二行(倒数第一行是“}”)，后边的这个逗号是一定要有的，不然网页中不会出现编辑器，冒号后对应汉语，是在网页中显示的内容。

2. 之后，在 editor/_source/internals/fckcommands.js 里新建一个 command:

   case ' TestMenu ':

   oCommand = new FCKTestMenuCommand ('TestMenu', FCKLang. TestMenu);

   break;

   注意那个红色字体的 **FCKTestMenuCommand** 了么?这个是一个自定义的 command，稍候将讨论它。

3. 然后，在 *editor/_source/internals/fckcontextmenu.js* 添加一个 context menu 。 在 case 'Generic' : 下面添加如下: oGroup.Add(new FCKContextMenuItem(this, 'TestMenu', FCKLang.TestMenu, true)) ; 其中第四个布尔类型的参数，如果指定为 true，说明菜单有图标，反之则无。在此，我指明了我所自定义的菜单是有图标的，那么我就应该把图标放在 editor\skins\xxx\toolbar 目录下，其中 xxx 就是你在你的配置文件里选择的编辑器的皮肤，但是我们的图标命名可是有学问的，必须是如 FCKContextMenuItem 构造器第二个字符串参数的小写字母形式，如，这里我的图标被命名为 TestMenu.gif 了。 Ok，如此右键菜单就被添加上去了，运行你的网页，可以看到此菜单了。

4. 然后，还有最关键的一步:用 editor 下的 fckeditor.original.html 网页的内容代替 fckeditor.html 的内容，同时不要忘记备份 fckeditor.html 文件!

5. 现在开始讨论步骤 2 中的 FCKTestMenuCommand。菜单添加上其了，你总得让其拥有什么功能吧?好，在 editor/_source/commandclasses/fck_othercommands.js 里仿照其他的命令新建一个对应的命令类。一般建立好类后，还要建立两个静态函数:xxx .prototype.Execute，xxx .prototype.GetState。前者是事件处理函数，即点击了该右键菜单所要执行的功能全部写在这个函数，后者返回菜单的状态。在此为了演示，我们实现一个最简单的功能: FCKTestMenuCommand.prototype.Execute = function(){ window.alert("It works!");}够简单吧，只是不痛不痒的弹出一个提示框。 当然，这里我们也可以不用自定义的 FCKTestMenuCommand，而直接用 FCKEditor 定义好的 command，比如 FCKDialogCommand，它也定义在 fck_othercommands.js 里，当然如果这样的话，弹出的会是一个 dialog。

5 文件上传问题

FCKeditor 集成 java servlet 可以实现文件的上传和服务器端列表读取功能 FCKeditor 自己提供了两个 servlet 来分别实现上传文件功能，和读取服务器端文件列表功能，这两个 servlet 分别为：

com.fredck.FCKeditor.connector.ConnectorServlet（读取文件列表）

com.fredck.FCKeditor.uploader.SimpleUploaderServlet（实现文件上传）

5.1 开启和关闭文件上传功能（fckconfig.js）

浏览上传：

```
FCKConfig.LinkBrowser = true ;
FCKConfig.ImageBrowser = true ;
FCKConfig.FlashBrowser = true ;
```

快速上传：

```
FCKConfig.LinkUpload = true ;
FCKConfig.ImageUpload = true ;
FCKConfig.FlashUpload = true ;
```

值为 `true` 是为功能开启，值为 `false` 时为功能关闭。

5.2 文件上传的基本使用

第一步： 在 web.xml 文件中增加如下代码：

<servlet>

```
    <servlet-name>Connector</servlet-name>
      <servlet-class>
        net.fckeditor.connector.ConnectorServlet
    </servlet-class>
     <init-param>
        <param-name>baseDir</param-name>
        <param-value>/UserFiles/</param-value>
  </init-param>
    <load-on-startup>1</load-on-startup>
  </servlet>

  <servlet-mapping>
    <servlet-name>Connector</servlet-name>
    <url-pattern>
      /fckeditor/editor/filemanager/connectors/*
    </url-pattern>
  </servlet-mapping>
```

意思是在浏览服务器上的 baseDir 配置指定里面的所有文件及其目录结构列表。如果你的 baseDir 没有配置，Connector 将会自动创建一个默认的文件夹 UserFiles，对应的 ConnectorServlet 中 init()方法中代码如下：

```
baseDir = getInitParameter("baseDir");
if (baseDir == null)
    baseDir = "/UserFiles/";
```

还想说一下的是，FCKeditor 的 client 调用 server 的 servlet 方法采用的是 Ajax 思想来实现。当你点击浏览服务器(browser server)的时候就会触发一个异步的 javascrīpt + xmlhttp 的调用响应，后台的 servlet 会去完成你要请求的事件，然后数据以 xml 方式返回给 client 来解析。很明显，你要实现去数据库或者其他的文件系统请求列表，你只要修改

ConnectorServlet 中两个私有方法：getFolders 和 getFiles

让它去你指定的地方得到文件列表即可，这样你的文件可以放在任何你指定目录下。多说一句，很多人都想知道个人 blog 系统中怎么实现上传文件以后对应用户浏览自己的列表的，我的做法很简单，建立你用户名的文件夹，你上传只能上传到你的目录夹，浏览可以通过程序指定浏览对应用户下的文件夹即可，这个时候你要修改 Connectorservlet 中的路径即可！

第二步：在类路径(src目录下)中创建一个 *fckeditor.properties* 文件，并在此文件中加上如下代码：

```
connector.userActionImpl=net.fckeditor.requestcycle.impl.UserActionImpl
```

第三步：将相关的 Jar 包 Copy 到 WebRoot\lib 目录下面。

5.3 上传中文文件名的文件会出现乱码

下载 fckeditor 的 java 源码：fckeditor-java-2.4-src.zip， 在 Eclipse 中将 fckeditor-java-core-2.4.jar 下的 net\fckeditor\connector\ConnectorServlet.class 文件打开，再自己新建一个类，如：*com.hmw.fck.ConnectorServlet.java*, 将 ConnectorServlet.class 中的代码 Copy 到此 Java 文件中，再在此类中 doPost 方法中的

```
try {
    List items = upload.parseRequest(request);
```

…… ……

前面加上一句 **upload.setHeaderEncoding("UTF-8");**

最后将 web.xml 文件中的

<servlet>

<servlet-name>Connector</servlet-name>

<servlet-class>

net.fckeditor.connector.ConnectorServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

修改为：

<servlet>

<servlet-name>Connector</servlet-name>

<servlet-class>

com.hmw.fck.ConnectorServlet

</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

重新发布，OK.

5.4 创建中文名目录会出现乱码

将上一节中创建的 *com.hmw.fck.ConnectorServlet.java* 类的 doGet 方法中的

```
String newFolderStr = UtilsFile.sanitizeFolderName(request
                        .getParameter("NewFolderName"));
logger.debug("Parameter NewFolderName: {}", newFolderStr);
File newFolder = new File(currentDir, newFolderStr);
```

第一句改为：

```
String tempStr = new
String(request.getParameter("NewFolderName").getBytes("ISO-8859-1"),"
UTF-8");
String newFolderStr = UtilsFile.sanitizeFolderName(tempStr);
```

5.5 引用中文名文件的图片不能正常显示

方法一(不推荐使用)：修改 %tomcat_home%\ conf\server.xml 文件：

在标签 <Connector port="8080" …… /> 中加上一个属性：**URIEncoding=”UTF-8”** 重新启动 Tomcat。

方法二：避免使用 中文 作为文件名。

在之前创建的 *com.hmw.fck.ConnectorServlet.java* 类的 doPost 方法中

```
String filename = FilenameUtils.getName(rawName);
String baseName = FilenameUtils.removeExtension(filename);
String extension = FilenameUtils.getExtension(filename);
```

的后面加上这样一行代码：

filename = UUID.randomUUID().toString() + "." + extension;

5.6 控件允许上传的文件的类型

设置允许上传的文件类型：

文 件分类	客户端验证配置(fckconfig.js)	服务器端验证配置(fckeditor.property)
File	FCKConfig.LinkUploadAllowedExtensions FCKConfig.LinkUploadDeniedExtensions	connector.resourceType.file.extensions.allowed
Image	FCKConfig.ImageUploadAllowedExtensions FCKConfig.ImageUploadDeniedExtensions	connector.resourceType.image.extensions.allowed
Flash	FCKConfig.FlashUploadAllowedExtensions FCKConfig.FlashUploadDeniedExtensions	connector.resourceType.flash.extensions.allowed
Media	无	connector.resourceType.media.extensions.allowed

注：

1. 客户端验证配置中的 …UploadAllowedExtensions 和 …UploadDeniedExtensions 不可同时使用。
2. 修改时客户端配置 和 服务器端的配置要一致。
3. 各扩展名之间用“|”分隔开，如：
   fckconfig.js 文件中 **FCKConfig.FlashUploadAllowedExtensions = ".(swf|flv)$" ;**
   fckeditor.property 文件中 **connector.resourceType.flash.extensions.allowed = swf|fla**

5.7 控制上传的文件的大小

做如下修改：

1. 在服务端的 Servlet 中在保存文件之前先判断一下文件的大小，如果超出限制就传递一个错误码，并不再保存文件。
2. 修改对应的页面中的回调函数，增加对这个自定义的错误码的处理。

如规定上传的文件不能大于 10K：

第一步： 在 *com.hmw.fck.ConnectorServlet.java* 文件的 doPost 方法中的

```
if (!ExtensionsHandler.isAllowed(resourceType, extension)) {
    ur = new UploadResponse(new Object[] { new Integer(202) });
}
```

后面加上如下代码：

```
else if(uplFile.getSize() > 1024 * 10){
    ur = new UploadResponse(new Object[] { new Integer(204) });
}
```

这里的 204 是由自己定义的，为其他数字也可以，但最好是不要与现在已经定义好了的错误码重复，要知道现在已经定义了哪些错误码，可在 *fckeditor\editor\dialog\fck_image\fck_image.js* 文件中的 OnUploadCompleted 方法中看到。

第二步： 在 *fckeditor\editor\dialog\fck_image\fck_image.js* 文件中的 OnUploadCompleted 方法中

```
case 203 :
    alert( "Security error. You probably don't have enough permissions
to upload. Please check your server." ) ;
    return ;
```

的后面加上如下代码：

```
case 204 :
    alert( "您上传的文件的大小超出了限制！" ) ;
    return ;
```

5.8 增加文件删除功能

在使用文件上传的过程中我们很容易发现一个问题，那就是上传后的文件我们却无法删除，这样会使网站上 的垃圾文件（主要是图片）日益增多，那我们是否能够通过对编辑器的修改能较为方便的管理删除无用上传文件呢？我们不难想到 fckeditor 的浏览文件功 能，对该功能进行适当的扩展，对日常所上传的文件进行管理就变得相当简单易行了，那我们赶紧动手吧：

增加该功能我们修改的页面为：*fckeditor\editor\filemanager\browser\default\frmresources list.html*

第一步，在页面的 body 区域增加如下代码：

```
<div id="showFile" style="float: left; display: none; background-color: #999999"></div>
<iframe id="iframe_del" name="iframe_del" width="0" height="0" scrolling="no"></iframe>
<div id="body_content"></div>
```

说明：第一对的作用是当我们把鼠标移动到所上传的文件上时显示其相关信息；<iframe></iframe>标签的作用是供我们删除上传文件时做动作响应窗口的，即是将我们的删除动作在 iframe 中运行，目的是实现伪无刷新删除效果；最后一对的作用是显示上传文件及文件夹的列表

第二步，在文件的 js 代码区中加入：

```
// 显示文件显示层
function showDiv(fileUrl) {
    var name = fileUrl;
    // 获取文件类型
    var suffix = name.substring(name.lastIndexOf(".") + 1);
    var div = document.getElementById("showFile");
    div.content = "";
    div.style.position = "absolute";
    div.content += "<table width='256' border='0' cellpadding='3' cellspacing='1' bgcolor='#737357'><tbody><tr><td height='23' align='left' bgcolor='#C7C78F'><table width='100%' height='100%' border='0' cellpadding='0' cellspacing='0'><tbody><tr><td width='47%'> <a href='javascript:' onClick='hiddenDiv();'><font color='#000000' style='text-decoration:none;'>关闭</font></a></td> <td width='53%' align='right'><a href='javascript:' onclick='delFile(\"" + fileUrl + "\");'><font color='#000000' style='text-decoration:none;'>删除</font></a> < /td></tr></tbody></table></td></tr>";
    if (suffix == 'gif' || suffix == "jpg" || suffix == "jpeg"
            || suffix == "bmp" || suffix == "png") {
        div.content += "<tr><td align='center' bgcolor='#C7C78F'><img src='" + fileUrl + "' onload='if(this.width>250) this.width=250'
```

```
style='margin:3px;'></td></tr>";
    } else {
        div.content += "<tr><td height='35' align='center'
bgcolor='#C7C78F'><strong>该类型不能预览</strong></td>< /tr>";
    }
    div.content += "</tbody></table>";
    div.innerHTML = div.content;
    div.style.display = "";
    div.style.top = event.y + document.body.scrollTop + 10;
    div.style.left = event.x + document.body.scrollLeft + 30;
}
// 隐藏文件显示层
function hiddenDiv() {
    var div = document.getElementById("showFile");
    div.style.display = "none";
}
// 在iframe中删除文件
function delFile(fileUrl) {
    if (!confirm('你确定删除该文件？'))
        return;
    var url = '/fckeditor/del_file.php?filePath=' + fileUrl;
    window.open(url, "iframe_del");
    Refresh();
}
```

第三步，修改本页面原有 js

将如下代码：

```
oListManager.Clear = function() {
    document.body.innerHTML = '';
}
```

修改为：

```
oListManager.Clear = function() {
    hiddenDiv();
    // body_content为我们在body区域增加的div标签
    document.getElementById("body_content").innerHTML = '';
}
```

找到：

```
var sLink = '<a href="#" onclick="OpenFile(\'' + fileUrl.replace(/'/g,
'\\\'') + '\');return false;">';
```

修改为：

```
// 即是文件信息显示功能
var sLink = '<a href="#" onmouseover="showDiv(\'' + fileUrl
        + '\');" onclick="OpenFile(\'' + fileUrl.replace(/'/g, '\\\'') +
'\');return false;">';
```

修改：

```
function Refresh() {
    LoadResources(oConnector.ResourceType, oConnector.CurrentFolder);
}
```

为：

```
function Refresh() {
    hiddenDiv(); // 所作修改，为了刷新列表时默认隐藏文件显示层
    LoadResources(oConnector.ResourceType, oConnector.CurrentFolder);
}
```

修改函数：GetFoldersAndFilesCallBack，找到：

```
document.body.innerHTML = oHtml.ToString();
```

修改为：

```
document.getElementById("body_content").innerHTML = oHtml.ToString();
```

最后，增加一个文件删除页面 del_file.php（该文件路径和增加的 js 函数 delFile(fileUrl)中的调用一致），作用为删除文件，给出成功与否的操作提示，参考代码：

```
[php]
<?php
$filePath = "..".trim($_GET['filePath']);
if ( $filePath ){
    @unlink($filePath);
    echo "<script>alert('删除成功。');</script>";
}else{
    echo "<script>alert('删除错误，可能文件不存在或者已经删除。');</script>";
}
?>
[/php]
```

6 超连接重定位问题

FCKeditor 可以插入超连接，实现对文件的预览功能，只要我们稍微改变我们可以使 FCKeditor 编辑器支持对任意文件系统下的任意文件的客户端浏览和下载！FCKeditor 本来提供的是相对 URL 超链接，只要我们修改 ConnectorServlet 中传递给客户端的地址的时候，把它改写成绝对 URL 然后再通过我们自己的 filter 的 servlet 实现重定向去一个下载/浏览文件的 struts 的 action 方法就可以实现在客户端对超连接文件的下载和浏览！具体做法如下：

第一步：修改 ConnectorServlet 传递给客户端 javascrīpt 的路径，代码如下：

String currentUrl = "http://" + request.getserver +request.getServerPort + request.getContextPath + resourcePath;

以上代码请在 ConnectorServlet 的 doGet()里面拼装！在调用 CreateCommonXml()私有方法的时候参数传入：

myEl.setAttribute("path",currentPath);

myEl.setAttribute("url",currentUrl);

提醒一下 resourcePath 为在 web.xml 配置文件中 ConnectorServlet 中的一个初始化参数配置，等一下利用 filter 实现对超连接的重定位就提取 URL 中的这个配置参数来判断，配置如下：

```
<init-param>
    <param-name>resourcePath</param-name>
    <param-value>/fileSystem/</param-value>
</init-param>
```

第二步：建立你的 filter servlet，实现对 URL 的截获，对符合要求的 URL 进行重定位到你的对应 action 中

去即可

第三步：实现你的对应 action 来实现文件的上传和下载功能即可！

第四步：扩展功能－实现对 URL 的加密，对连接的 URL 中加上一串字符，最后几位作为算法校验，对不符合要求的 URL 连接,filter 将会拒绝重定位到指定 action。此外利用自己写的扩展类还可以实现对超连接的文件类型进行限制，比如你只能超连接 JPG|GIF|DOC|TXT|HTML 等几种后缀名的文件，对其他文件即使你指定超连接也让你浏览和下载，这些都可以在 web.xml 中通过修改对应 servlet 的配置文件的初始化参数实现。

7 使用 FCKeditor 的 API

FCK 编辑器加载后，将会注册一个全局的 FCKeditorAPI 对象。FCKeditorAPI 对象在页面加载期间是无效的，直到页面加载完成。如果需要交互式地知道 FCK 编辑器已经加载完成，可使用“FCKeditor_OnComplete”函数。

```
<script type="text/javascript">
  function FCKeditor_OnComplete(editorInstance) {
    FCKeditorAPI.GetInstance('FCKeditor1').Commands.GetCommand('FitWi
ndow').Execute();
  }
</script>
```

FCKeditor 编辑器，提供了非常丰富的 API，用于给 End User 实现很多想要定制的功能，比如最基本的数据验证，如何在提交的时候用 JS 判断当前编辑器区域内是否有内容，FCK 的 API 提供了 GetLength()方法；再比如如何通过脚本向 FCK 里插入内容，使用 InsertHTML()等；还有，在用户定制功能时，中间步骤可能要执行 FCK 的一些内嵌操作，那就用 ExecuteCommand()方法。详细的 API 列表，请查看 FCKeditor 的 Wiki(http://wiki.fckeditor.com)。而常用的 API，请查看 FCK 压缩包里的_samples/html/sample08.html。此处简单的列出了几个：

7.1 获得 FCKeditor 的实例

7.1.1 获得当前页 FCKeditor 实例

```
var oEditor = FCKeditorAPI.GetInstance('InstanceName');
```

7.1.2 从 FCKeditor 的弹出窗口中获得 FCKeditor 实例

```
var oEditor = window.parent.InnerDialogLoaded().FCK;
```

7.1.3 从框架页面的子框架中获得其它子框架的 FCKeditor 实例

```
var oEditor =
window.FrameName.FCKeditorAPI.GetInstance('InstanceName');
```

7.1.4 从页面弹出窗口中获得父窗口的 FCKeditor 实例

```
var oEditor = opener.FCKeditorAPI.GetInstance('InstanceName');
```

7.2 常见的 Js 方法调用

7.2.1 插入 HTML 到 FCKeditor

```
function insertHTMLToEditor(codeStr){
 var oEditor = FCKeditorAPI.GetInstance("FCKeditor1");
 if (oEditor.EditMode==FCK_EDITMODE_WYSIWYG){
   oEditor.InsertHtml(codeStr);
 }else{
   return false;
 }
}
```

7.2.2 设置 FCKeditor 的内容(HTML)

```
function SetContents(contentStr)
{
    // Get the editor instance that we want to interact with.
    var oEditor = FCKeditorAPI.GetInstance('FCKeditor1') ;
    // Set the editor contents (replace the actual one).
    oEditor.SetData(contentStr) ;
//或者：oEditor.SetHTML("content", false); // 第二个参数：true|false，是否以所见即所得方式设置其内容（此参数可写也可不写）
}
```

7.2.3 获取 FCKeditor 中的 XHTML

```
function GetContents()
{
    // Get the editor instance that we want to interact with.
    var oEditor = FCKeditorAPI.GetInstance('FCKeditor1') ;
    // Get the editor contents in XHTML.
    return oEditor.GetXHTML(true) ; //"true" means you want it formatted.
//或者直接为： return oEditor.GetXHTML() ;
}
```

7.2.4 获取 FCKeditor 中的 innerHTML 和 innerText

```
function GetInnerHTML()
{
    var oEditor = FCKeditorAPI.GetInstance('FCKeditor1') ;
    return oEditor.EditorDocument.body.innerHTML ;
//获取文字内容为：return oEditor.EditorDocument.body.innerText ;
}
```

7.2.5 执行指定动作

```
function ExecuteCommand(commandName){
 var oEditor = FCKeditorAPI.GetInstance("FCKeditor1") ;
 oEditor.Commands.GetCommand(commandName).Execute() ;
}
```

注：ExecuteCommand函数可在 FCK 编辑器之外调用 FCK 编辑器工具条命令：

命令参数列表如下：

命令参数	功能	命令参数	功能
Source	源代码	DocProps	页面属性
-	丨分隔符	Save	保存
NewPage	新建	Preview	预览
Templates	模板	Cut	剪切
Copy	复制	Paste	粘贴
PasteText	粘贴为无格式文本	PasteWord	从 MS Word 粘贴
Print	打印	SpellCheck	拼写检查
Undo	撤消	Redo	重做
Find	查找	Replace	替换
SelectAll	全选	RemoveFormat	清除格式
Form	表单	Checkbox	复选框
Radio	单选框	TextField	单行文本
Textarea	多行文本	Select	列表菜单
Button	按钮	ImageButton	图像域
HiddenField	隐藏域	Bold	加粗
Italic	倾斜	Underline	下划线
StrikeThrough	删除线	Subscript	下标
Superscript	上标	OrderedList	插入／删除编号列表
UnorderedList	插入／删除项目列表	Outdent	减少缩进
Indent	增加缩进	JustifyLeft	左对齐
JustifyCenter	居中对齐	JustifyRight	右对齐
JustifyFull	两端对齐	Link	插入／编辑链接
Unlink	取消链接	Anchor	插入／编辑锚点链接
Image	插入编辑图像	Flash	插入／编辑 Flash
Table	插入／编辑表格	Rule	插入水平线
Smiley	插入表情	SpecialChar	插入特殊符号
PageBreak	插入分页	Style	样式
FontFormat	格式	FontName	字体
FontSize	大小	TextColor	文本颜色
BGColor	背景颜色	FitWindow	全屏编辑
About	关于 Fuckeditor		

如：

```
<input type="button" name="Submit" value="预览"
```

```
onClick="javascript:ExecuteCommand('Preview')">
```

7.2.6 统计编辑器中内容的字数

```
function getLength(){
 var oEditor = FCKeditorAPI.GetInstance("FCKeditor1");
 var oDOM = oEditor.EditorDocument;
 var iLength ;
 if(document.all){
   iLength = oDOM.body.innerText.length;
 }else{
   var r = oDOM.createRange();
   r.selectNodeFCKeditor1s(oDOM.body);
   iLength = r.toString().length;
 }
 alert(iLength);
}
```

7.2.7 检查 FCKeditor 中的内容是否有改动

```
function CheckIsDirty()
{
    var oEditor = FCKeditorAPI.GetInstance('FCKeditor1') ;
    return oEditor.IsDirty() ;
}
```

7.2.8 将 FCKeditor 中的内容是否有改动的值重新设置

```
function ResetIsDirty()
{
    var oEditor = FCKeditorAPI.GetInstance('FCKeditor1') ;
    oEditor.ResetIsDirty() ;
}
```

8 外联编辑条(多个编辑域共用一个编辑条)

这个功能是 2.3 版本才开始提供的，以前版本的 FCKeditor 要在同一个页面里用多个编辑器的话，得一个个创建，现在有了这个外联功能，就不用那么麻烦了，只需要把工具条放在一个适当的位置，后面就可以无限制的创建编辑域了。

要实现这种功能呢，需要先在页面中定义一个工具条的容器： ，然后再根据这个容器的 id 属性进行设置。

JAVASCRIPT 实现代码：

```
<div id="xToolbar"></div>
```

创建 FCKeditor 1: 和 FCKeditor 2:

```
<script type="text/javascript">
    // Automatically calculates the editor base path based on the _samples
directory.
```

```
    // This is usefull only for these samples. A real application should
use something like this:
    // oFCKeditor.BasePath = '/fckeditor/'; // '/fckeditor/' is the
default value.
    var sBasePath =
document.location.pathname.substring(0,document.location.pathname.las
tIndexOf('_samples'));

    var oFCKeditor = new FCKeditor( 'FCKeditor_1' );
    oFCKeditor.BasePath = sBasePath;
    oFCKeditor.Height = 100;
    oFCKeditor.Config[ 'ToolbarLocation' ] = 'Out:parent(xToolbar)';
    oFCKeditor.Value = 'This is some <strong>sample text</strong>. You
are using FCKeditor.';
    oFCKeditor.Create();

    oFCKeditor = new FCKeditor( 'FCKeditor_2' );
    oFCKeditor.BasePath = sBasePath;
    oFCKeditor.Height = 100;
    oFCKeditor.Config[ 'ToolbarLocation' ] = 'Out:parent(xToolbar)';
    oFCKeditor.Value = 'This is some <strong>sample text</strong>. You
are using FCKeditor.';
    oFCKeditor.Create();
</script>
```

此部分的详细 DEMO 请参照*_samples/html/sample11.html*，*_samples/html/sample11_frame.html*

9 解释 fck 样式(CSS)的工作原理

fck 的样式设置涉及到了两个文件，一个是你定义好的样式表文件.css，另一个是告诉 fck 样式表如何使用的 xml 文件，两个文件缺一不可。css 文件的位置是不做要求的，但是需要你在应用的编辑器的页面上插入样式表文件的链接。这样才能显示出来样式。fckstyles.xml 在与 editor 目录同级的目录下。该文件定义了那些样式可以使用在那些标签里面。

下面是 fck 自带的样式 xml 定义中的一部分：

```
<Style name="Image on Left" element="img">
    <Attribute name="style" value="padding: 5px; margin-right:5px" />
    <Attribute name="border" value="2" />
    <Attribute name="align" value="left" />
</Style>
<Style name="Image on Right" element="img">
    <Attribute name="style" value="padding: 5px; margin-left: 5px" />
    <Attribute name="border" value="2" />
    <Attribute name="align" value="right" />
</Style>
```

每一个<style>将来会生成一个样式的菜单项。name 名称就是显示在菜单里的文字；element 定义了该样式可以应用在那种 html 标签上，<Attribute>的 name 指定了将会修改标签的哪个属性来应用样式，value

则是修改成的值。

看这个：

```
<Style name="Title" element="span">
    <Attribute name="class" value="Title" />
</Style>
```

如果你在 fck 选定了文字 “FCKeditor 实战技巧”并应用该样式 则原来文字就会变成** FCKeditor 实战技巧**

注意：如果编辑器呈整页编辑状态，那么整页里面也需要插入样式表链接才能显示出来样式。

10 获取 FCKeditor 中插入的图片

下面的代码为当 FCKeditor 失去焦点时将 FCKeditor 中所有的图片的地址显示到 img_select 下拉框中：

```
<script type="text/javascript">
    function FCKeditor_OnComplete( editorInstance )
    {
        //FCKeditor加载完成后给其注册一个失去焦点事件
        editorInstance.Events.AttachEvent( 'OnBlur', onEditorBlur ) ;
    }

    function onEditorBlur(){
        var oSelect = $("img_select");
        for(var i=oSelect.options.length-1; i>0; i--){
            oSelect.options[i] = null;
        }
        oEditor = FCKeditorAPI.GetInstance('EditorDefault');
        var imgs = oEditor.EditorDocument.body.all.tags("img");
         for(var i=0; i < imgs.length; i++){
            var oOption = document.createElement("option");
            oOption.appendChild(document.createTextNode(imgs[i].src));
            oSelect.appendChild(oOption);
        }
    }
</script>
```

2008 年 10 月 15 日